Supporting material for “Incorporating sequence quality data into alignment improves DNA read mapping”

Martin C. Frith, Raymond Wan and Paul Horton

Computational Biology Research Center,

Institute for Advanced Industrial Science and Technology,

2-4-7 Aomi, Koto-ku, Tokyo 135-0064, Japan

1 Calculation of alignment scores

The main text describes an alignment scoring scheme that incorporates sequence quality data: here we describe how our software performs these calculations. The input given to the algorithm includes a scoring matrix, and read quality scores in either FASTQ format (one quality score per base) or PRB format (four quality scores per base). In order to use the scoring scheme in equation 12 of the main text, we need to pre-calculate T, R_{xy}, and $P(y|d)$. Given a scoring matrix, it is possible to calculate T uniquely [5], and we do so using software kindly supplied by Yu and Altschul. With T in hand, R_{xy} is simply $exp(S_{xy}/T)$. We convert quality score Q_y to $P(y|d)$ as follows:

$$P(y|d) = 1 - 10^{-Q_y/10} \tag{S1}$$

or:

$$P(y|d) = 1/(1 + 10^{-Q_y/10})\,, \tag{S2}$$

depending on the type of quality score provided. For FASTQ data, we estimate the probability of the most probable base m as above and the probabilities for the other three bases as:

$$P(z|d)_{z \neq m} = (1 - P(m|d))\,/3\,. \tag{S3}$$

To gain speed, we pre-calculate lookup tables, so that every possible quality score can be immediately translated to $P(y|d)$ and $P(z|d)$. This is feasible because quality scores are always small integers.

Calculation of the position-specific-scoring-matrix using:

$$S'_{xd} = T \times \ln \sum_y \left(R_{xy} P(y|d) \right) \tag{S4}$$

is still a little slow. For simple scoring matrices in which all matches have the same score $S_{\text{match}} = T \times \ln R_{\text{match}}$, and all mismatches have the same score $S_{\text{mismatch}} = T \times \ln R_{\text{mismatch}}$, we use a faster method. In this case, noting that:

$$\sum_{y \neq x} P(y|d) \;=\; 1 - P(x|d) \tag{S5}$$

we observe that the following is equivalent to equation S4:

$$S'_{xd} \;=\; T \times \ln \left[\, P(x|d) \times R_{\text{match}} + (\, 1 - P(x|d)\,) \times R_{\text{mismatch}} \,\right] . \tag{S6}$$

Since this computation depends only on $P(x|d)$, which is stipulated by Q_x, we can precompute lookup tables for translating quality scores directly to S'_{xd}. (For PRB data, this formula may give numerically different results than equation S4. This is because the quality scores in the PRB file have each been rounded to the nearest integer, and after rounding errors the probabilities generally do not sum to exactly 1.)

In our software, the values S'_{xd} in the position-specific-scoring-matrix are rounded to the nearest integer.

2 Mapping probabilities for local and semi-global alignment

In our formulation a read mapping is simply an alignment between the read and the genome. We motivate the probability formula for a mapping μ,

$$p(\mu) = e^{S_\mu/T} / \sum_{\alpha \in \text{mappings}} \left(e^{S_\alpha/T} \right) , \tag{S7}$$

based on a probabilistic model of alignment. We consider both local and semi-global alignment. “Local” alignment allows alignment between any part of the read and any part of the genome; while “semi-global” alignment requires the entire read to align. In either case, the probability of a mapping μ can be written as the following Bayesian formula:

$$P(\mu|\text{data}) = \frac{P(\text{data}|\mu) \times P(\mu)}{\sum_{\alpha \in \text{mappings}} (P(\text{data}|\alpha) \times P(\alpha))} , \tag{S8}$$

where the data consists of the genome, the read and its quality scores. The probability of the observed data (the genome sequence and the sequencer read) given a mapping α is:

$$P(\text{data}|\alpha) = \prod_{l \in \text{aligned in } \alpha} P(x_l d_l|A) \;\times \prod_{j \in \text{genome unaligned in } \alpha} P(x_j) \;\times \prod_{k \in \text{read unaligned in } \alpha} P(d_k) \tag{S9}$$

Where x_l denotes the lth aligned genome base and d_l the sequencer data for the read position aligned to it, x_j denotes the jth unaligned genome base, and d_k denotes the sequencer data for the kth unaligned position of the read.

This formula can be rearranged to obtain:

$$\begin{aligned} P(\text{data}|\alpha) = & \prod_{l \,\in\, \text{aligned in } \alpha} P(x_l d_l|A) \;\times \prod_{j \,\in\, \text{genome unaligned in } \alpha} P(x_j) \;\times \prod_{k \,\in\, \text{read unaligned in } \alpha} P(d_k) \\ = & \; B \prod_{l \,\in\, \text{aligned in } \alpha} (\, P(x_l d_l|A) / P(x_l) P(d_l) \,) \\ = & \; B \prod_{l \,\in\, \text{aligned in } \alpha} R'(x_l d_l) \end{aligned} \tag{S10}$$

where,

$$B \;=\; \prod_{i \,\in\, \text{entire genome}} P(x_i) \;\;\times\; \prod_{j \,\in\, \text{entire read}} P(d_j)\,. \tag{S11}$$

Since B does not depend on the alignment, the probability of a mapping given the data (equation S8) can be rewritten:

$$P(\mu|\text{data}) = \frac{P(\mu)\prod_{l \,\in\, \text{aligned in}\, \mu} R'(x_l d_l)}{\sum_{\alpha\in\text{mappings}}\Big(P(\alpha)\prod_{l \,\in\, \text{aligned in}\, \alpha} R'(x_l d_l)\Big)} \tag{S12}$$

This is equivalent to equation S7.

Semi-global alignment is theoretically more sensitive than local alignment, in cases where its model is appropriate (i.e. we are certain that the whole read comes from a contiguous region of the genome). This may not always be true. Moreover, the fact that BLAST-like alignment tools such as LAST (and BLAST itself of course) employ heuristics to improve computation speed and may not always report all high scoring mappings, complicates the issue. Thus it is not clear *a priori* that semi-global alignment will outperform local alignment in practice.

2.1 Comparison to previous mapping probability calculations

The calculation described above is similar to ones described previously [1, 3]. The main novelties are that we model real sequence differences, using a scoring matrix and gap costs, and we allow for local alignment. If these ingredients are removed (e.g. by setting the mismatch and gap costs to infinity), our calculation becomes equivalent to the previous ones.

Li et al. describe three sources of mapping error [3]:

Type-1 Spuriously mapped reads that do not really come from the reference genome sequence at all.

Type-2 Erroneously mapped reads, whose true mapping to the reference was missed by the heuristic alignment algorithm.

Type-3 Erroneously mapped reads, reads whose true mapping was not missed by the algorithm.

Our calculation accounts for type-3 errors only. In the work of Li et al., type-1 errors are “not counted”, but they do give a formula for type-2 errors. In more recent work by these authors, however, they “assume the true hit can always be found” [2]. We understand this to mean that they no longer account for type-2 errors.

3 Justifications for some mapping parameters

In this study, we needed to fix values for various parameters such as the mapping probability threshold and the spaced seed pattern. Here, we justify some of our choices.

3.1 Mapping probability threshold

Throughout this study, we used alignments with mapping probability $\geq$ 0.99. Figure S4 shows what happens if we instead use 0.9 or 0.999. With a threshold of 0.9, we get significantly more false mappings. With a threshold of 0.999, we initially get fewer false mappings, but the number of correct mappings ultimately found is slightly lower. The "right" mapping probability threshold depends on the balance we wish to achieve between the number of correct and incorrect mappings, but these findings show that 0.99 appears reasonable.

3.2 Spaced seed pattern

`LAST` can use "adaptive spaced seeds", which we explain with an example. Suppose we use this spaced seed pattern: `1111110`. Starting from a particular base in a DNA read, `LAST` will look for the shortest sequence that occurs no more than (say) ten times in the genome. However, it will tolerate mismatches at every seventh base (the pattern is cyclically repeated).

We have not rigorously determined which spaced seed pattern is optimal. Here, we just tried three patterns: `1` (i.e. ordinary non-spaced seeds), `1111110`, and `11111011000`. The reason we tried the latter two is that these (when cyclically repeated out to a sufficient size) are optimal for guaranteeing to find alignments with, respectively, one or two mismatches.

For simulated reads, spaced seeds give clearly better mapping accuracy than non-spaced seeds (Figure S5). This is consistent with previous observations that spaced seeds are superior [4]. The pattern `1111110` seems slightly better than `11111011000` for low levels of substitution (0.2% to 1%), but it might become worse for higher levels of substitution. In any case, we selected `1111110` as the default seed for this study.

3.3 Scale for the score parameters

The score parameters shown in Table 1 of the main text involve an arbitrary scale factor, T. We set the scale by fixing the match score to be exactly 6 in all cases. This is not necessary, but we did so because it causes T to be approximately $10/\ln(10)$, and thus the resulting alignment scores are roughly comparable to phred scores.

3.4 Local versus semi-global alignment

We had to choose whether to align reads to the genome using local or semi-global alignment. For our tests with simulated reads, we might expect semi-global alignment to work better, because by design the entire read comes from the genome. For real data, this may not be true: in our experience, reads often have contaminating sequence at either or both ends.

`LAST` (like `BLAST`) is designed to do local alignment, but it can be tricked into doing semi-global alignment. The trick is simply to add 100 to each entry in the scoring matrix, so that all matches and mismatches have positive scores. After alignment, we subtract $100 \times (\text{read length})$ from the alignment scores. `LAST` will refuse to calculate T for such a scoring matrix, but we can supply T manually. This trick would need to be modified for alignment with gaps, but we only used gapless semi-global alignment in this study.

Mapping accuracy for simulated reads with local and semi-global alignment is shown in Figure S6. Surprisingly, local alignment does rather well; producing fewer false mappings initially, although it ultimately finds slightly fewer correct mappings than semi-global alignment. On the other hand, if we do the mapping in two-mismatch-guarantee mode, the advantage of local alignment almost vanishes (Figure S7). This can be explained as follows: if a read aligns to several genome locations, the difference in alignment scores will tend to be lower with local alignment, because local alignment allows more flexibility. Thus, local alignment will produce mapping probabilities $\geq$ 0.99 less often than semi-global alignment. This makes it more essential not to miss any high-scoring alignments when doing semi-global alignment. In other words, local alignment is more robust to heuristic algorithms that miss some alignments.

4 Gap costs for aligning *D. melanogaster* and *D. simulans* DNA

As mentioned in the main text, we obtained suitable gap costs by examining genome alignments of *D. melanogaster* versus *D. simulans*. We first measured the gap frequency (one per 101 aligned bases), and average gap size (6.67). We then calculated the following probabilities:

$$p(\text{open a gap}) = 1/(\text{alignedBasesPerGap} + 1) \tag{S13}$$
$$p(\text{extend a gap}) = (\text{meanGapSize} - 1)/\text{meanGapSize} \tag{S14}$$
$$p(\text{close a gap}) = 1/\text{meanGapSize}\,. \tag{S15}$$

The gap extension penalty is:

$$\text{GEP} = -T\ln[p(\text{extend a gap})]\,. \tag{S16}$$

The gap opening penalty is:

$$\text{GOP} = -T\ln[p(\text{open a gap}) \times p(\text{close a gap})/2] - \text{GEP}\,. \tag{S17}$$

The factor of 2 appears because the gap may occur in either of the two sequences. We subtract GEP in order to arrange that the gap cost is GOP $+$ GEP $\times$ (gap length), rather than GOP $+$ GEP $\times$ (gap length $- 1$).

5 Scalability and memory requirements of LAST

LAST currently uses 5-6 bytes of memory per base in the genome. It can handle a 3-billion-bp mammalian genome using about 16 gigabytes of memory. Alternatively, it can handle a mammalian genome in 2 gigabytes, by automatically grouping the chromosomes into sufficiently small groups, and aligning the query sequences to each group in turn. This grouping approach is slower, however. So there is a trade-off between speed and memory usage, which can be adjusted by the user.

We are investigating other techniques to reduce the memory usage (e.g. compressed suffix array, FM-index). In our understanding, these techniques have a practical cost in terms of speed. Our first attempt to use a compressed suffix array proved slower than chromosome-grouping (Kengo Sato, unpublished results). So the value of these techniques is not yet clear.

6 Supporting Figures

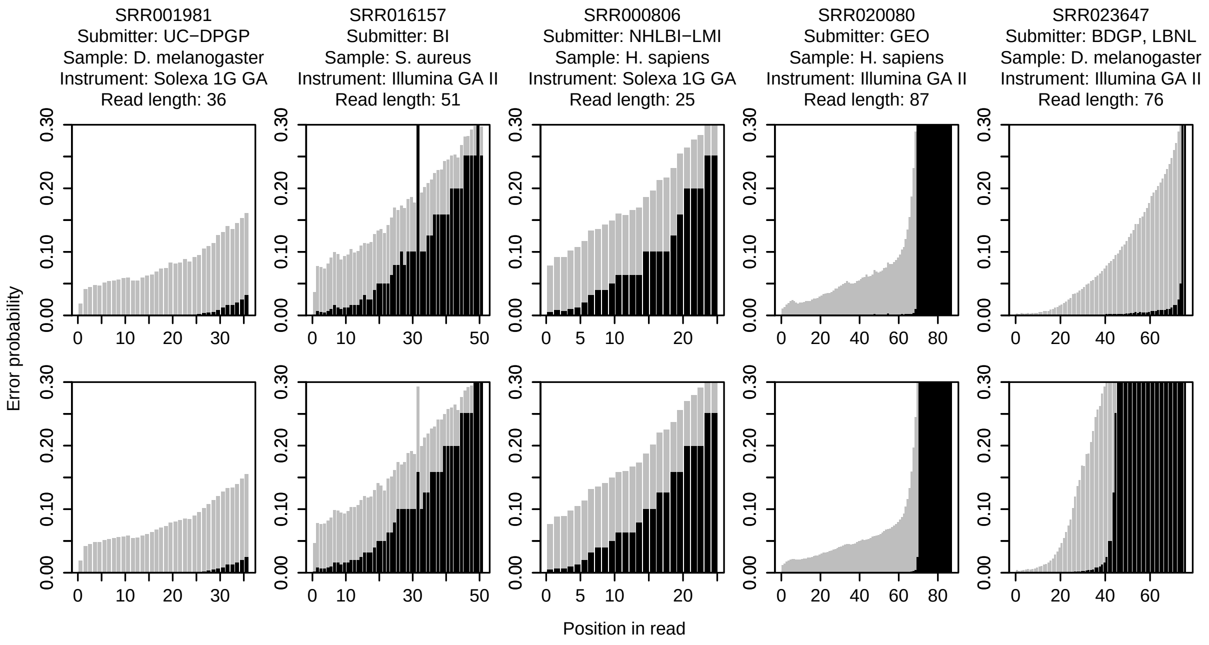

Figure S1: Estimated error rates for several DNA short-read datasets, from different submitters. The datasets and title information were obtained from the NCBI Short Read Archive. Each black bar shows the median error rate at one position, among 100,000 DNA reads. Each gray bar shows the mean error rate. These datasets were picked at random, without (e.g.) preferring datasets with high error rates. These datasets show different error patterns, but the datasets used in this study (first two columns) do not have particularly unusual error rates. The graphs in the top row used the first 100,000 reads in each dataset, whereas the graphs in the bottom row used 100,000 randomly chosen reads from each dataset. Although this does affect the distributions, it does not greatly alter the error rates of the two datasets used in this study.

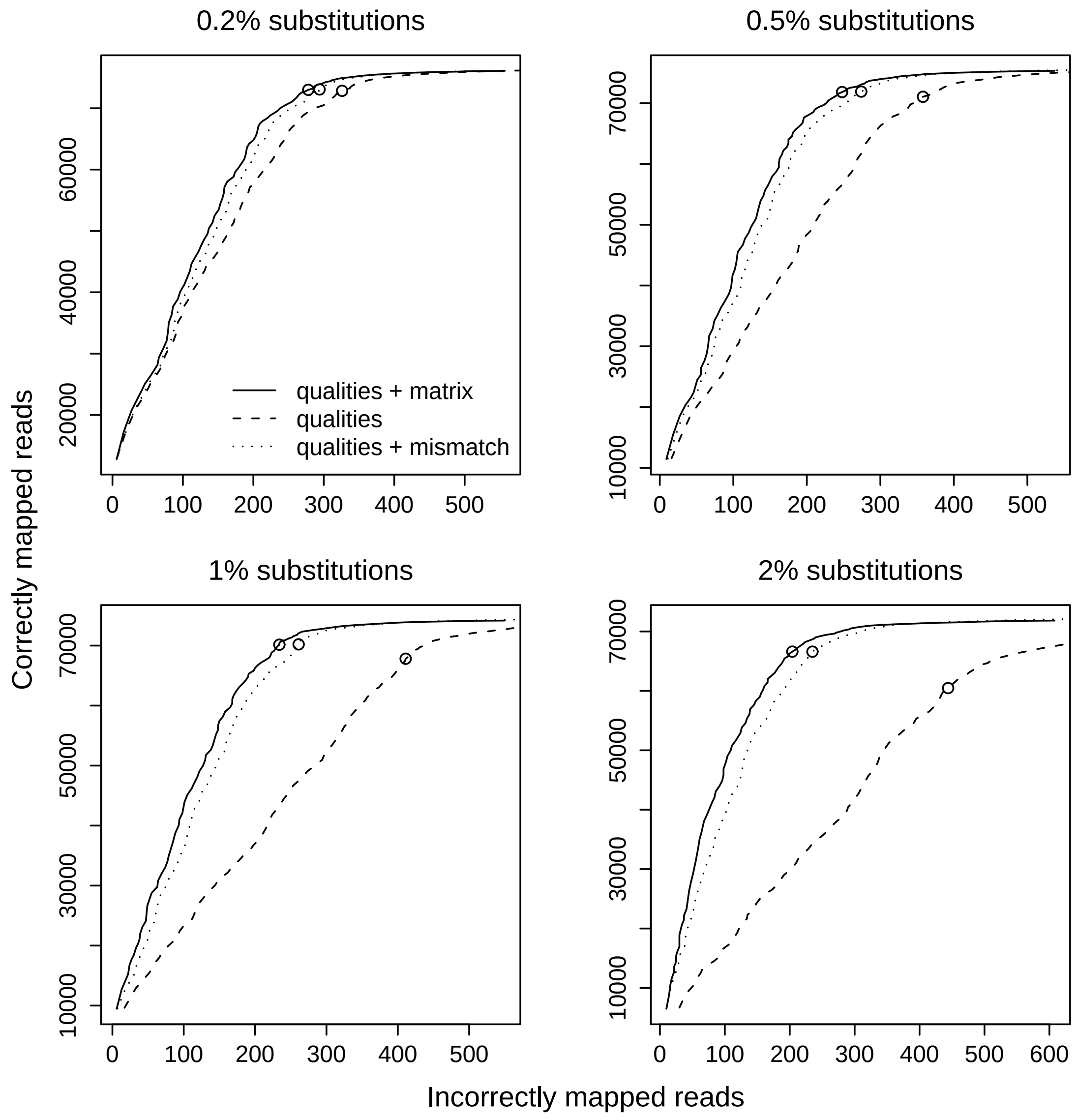

Figure S2: Mapping accuracy for 100,000 simulated 36-bp reads. The reads differ from the genome by a certain rate of “real” substitutions (0.2%, 0.5%, 2%, or 5%), plus sequencer errors. In each case, 60% of the “real” substitutions are transitions and 40% are transversions. Each line shows the relationship between the number of correctly and incorrectly mapped reads as the alignment score cutoff is varied. Circles indicate a score cutoff of 150. Dashed lines show the mapping accuracy when we model the sequencer errors but not the substitutions. Solid lines show the accuracy when we model both. Dotted lines show the accuracy when we model both, but ignore the difference between transitions and transversions. The solid and dashed lines here are identical to those in Figure 2 of the main text.

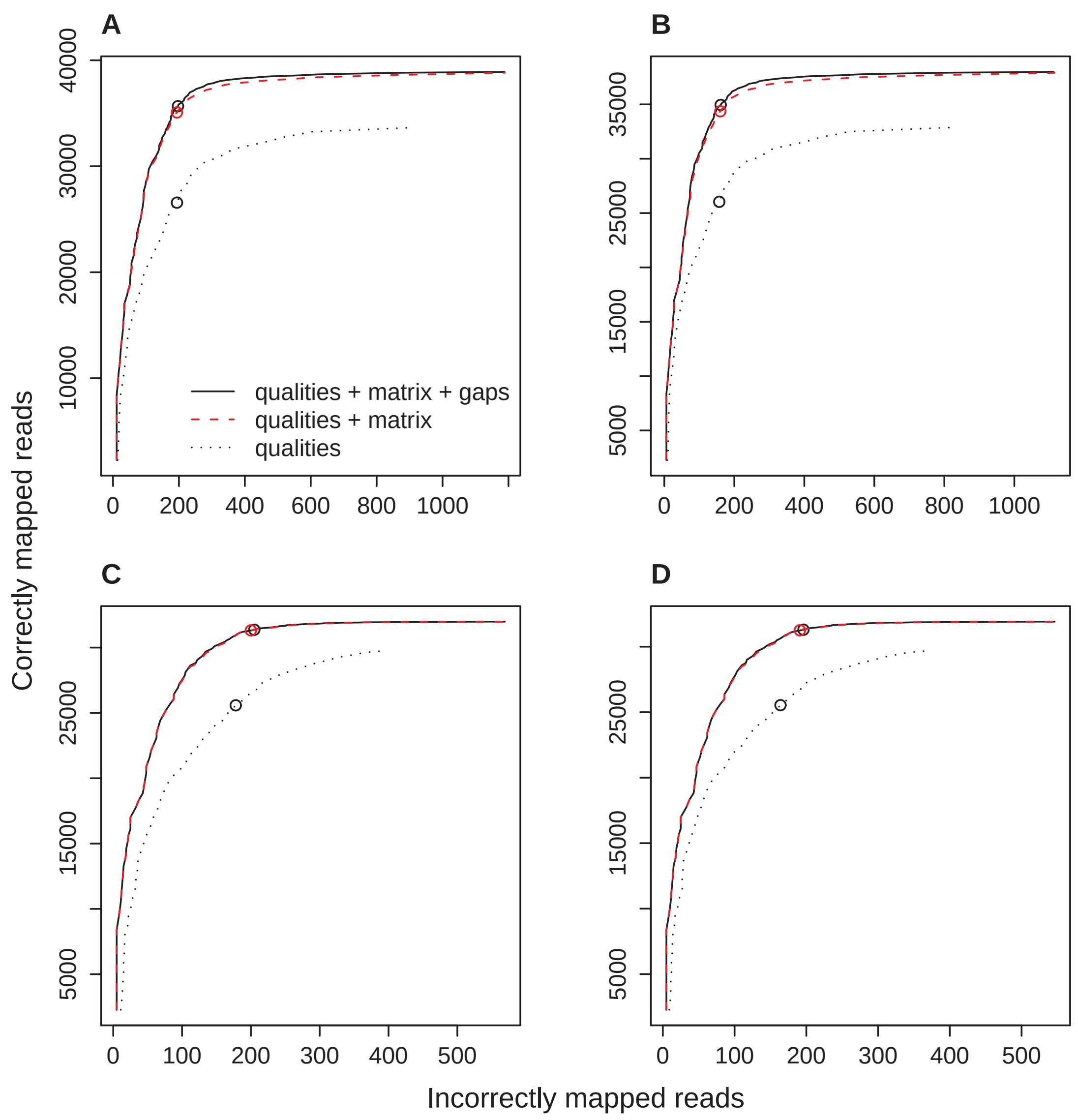

Figure S3: Estimated mapping accuracy for 100,000 real 36-bp reads from *D. melanogaster*, mapped to the *D. simulans* genome. Circles indicate a score cutoff of 150. Dotted lines show the mapping accuracy when we model the sequencer errors but not the real differences. Solid lines show the accuracy when we model both. Dashed red lines show the accuracy when we model both but forbid insertions and deletions. Correctness was estimated by mapping the reads to the *D. melanogaster* genome (modeling sequencer errors only), and using the UCSC *D. melanogaster*/*D. simulans* pairwise genome alignment to cross-reference the mappings. (A) The reads were mapped to both genomes using the default mode of LAST. This panel is identical to Figure 5 in the main text. (B) The reads were mapped to *D. simulans* in default mode, but to *D. melanogaster* in two-mismatch-guarantee mode. (C) The reads were mapped to *D. simulans* in two-mismatch-guarantee mode and to *D. melanogaster* in default mode. (D) The reads were mapped to both genomes in two-mismatch-guarantee mode.

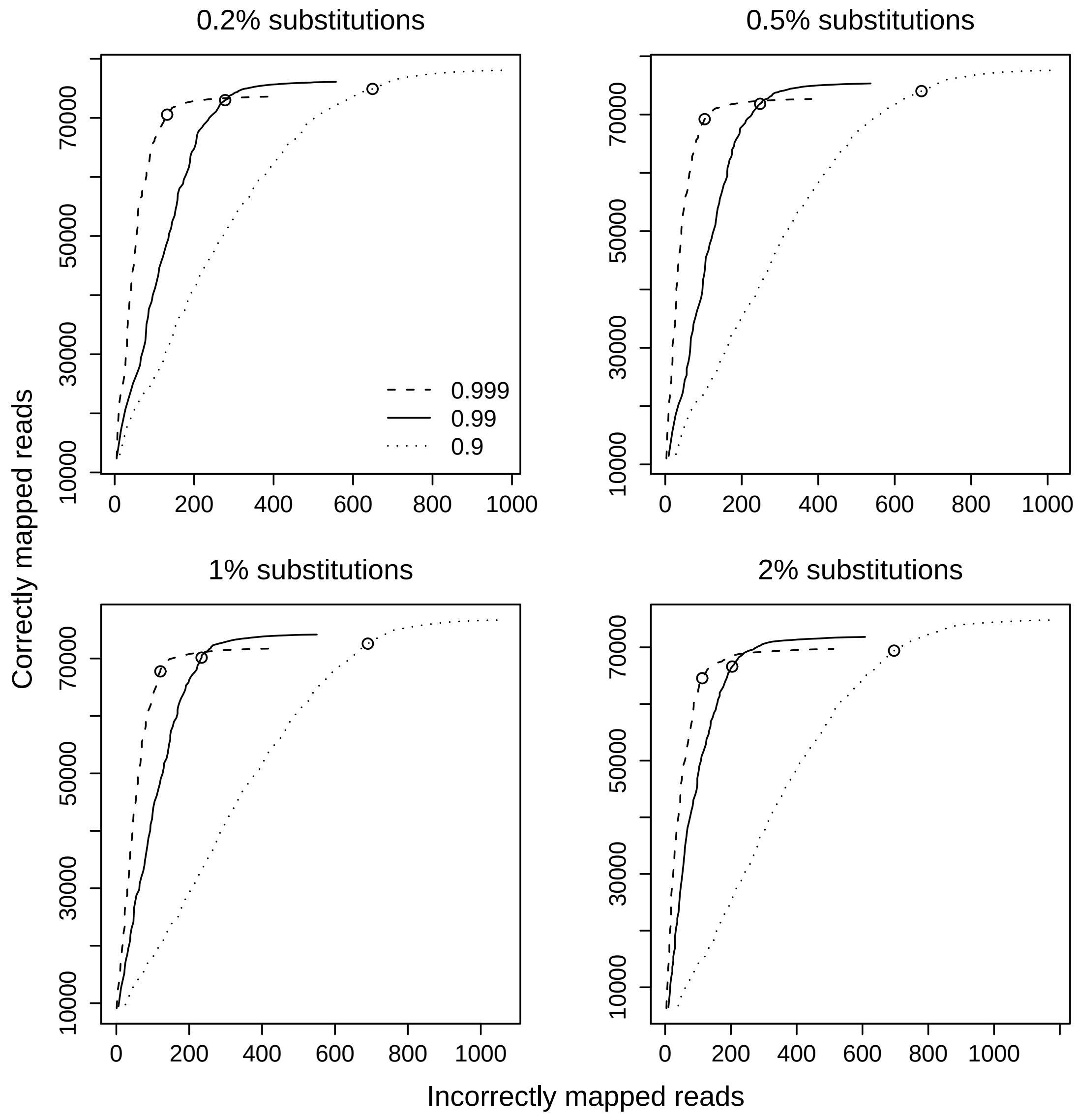

Figure S4: Mapping accuracy for 100,000 simulated 36-bp reads, with different mapping probability thresholds. Solid lines show the accuracy when we use mappings with probability $\geq$ 0.99. Dashed lines show the accuracy for mapping probability $\geq$ 0.999. Dotted lines show the accuracy for mapping probability $\geq$ 0.9. In all cases, we modeled both the sequencer errors and the "real" substitutions. The other figures all use a threshold of 0.99: so the solid lines here are identical to those in Figures 2, S2, S5, and S6.

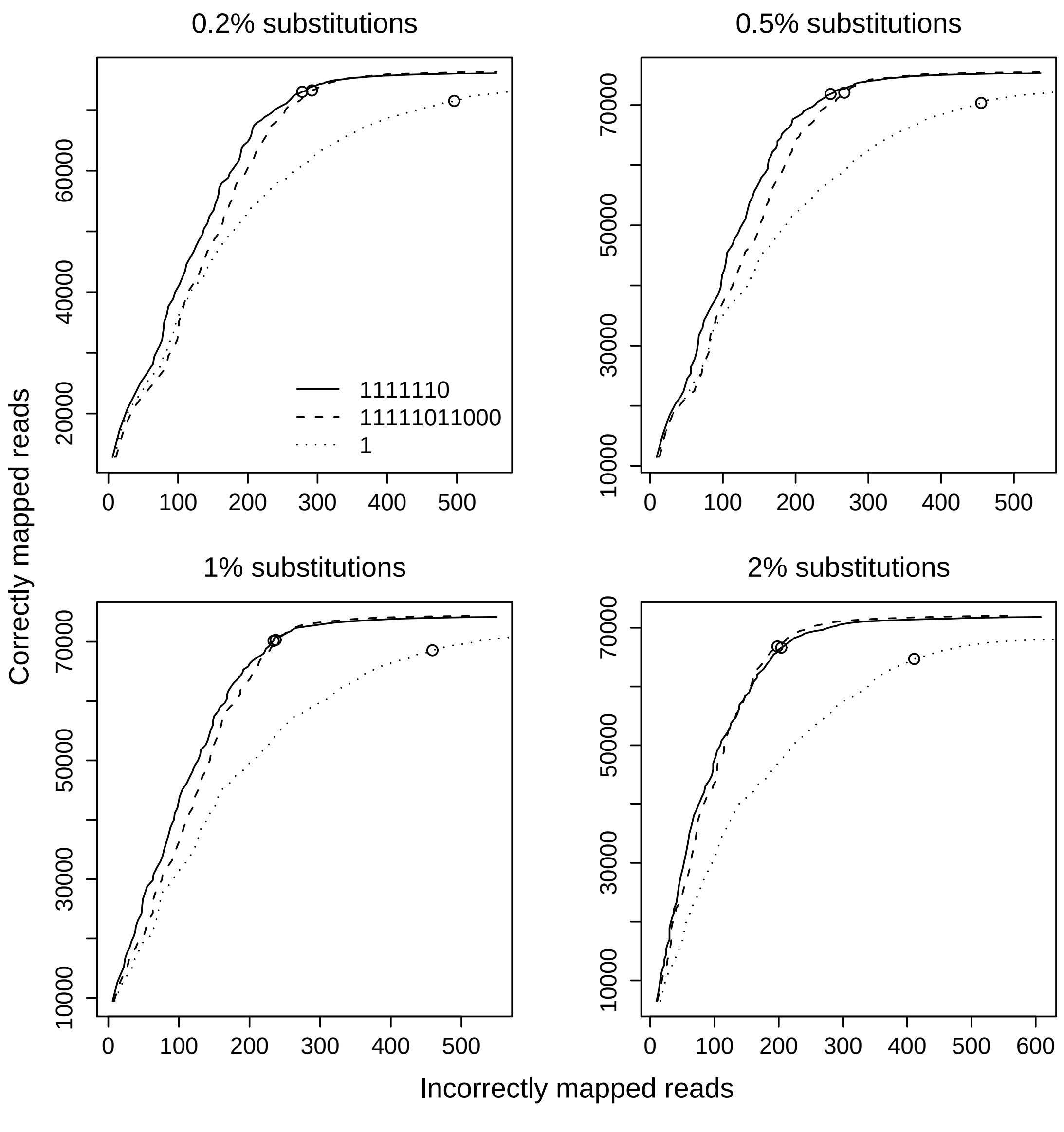

Figure S5: Mapping accuracy for 100,000 simulated 36-bp reads using different spaced seed patterns. In all cases, we modeled both the sequencer errors and the "real" substitutions. The solid lines here are identical to those in Figures 2, S2, S4, and S6.

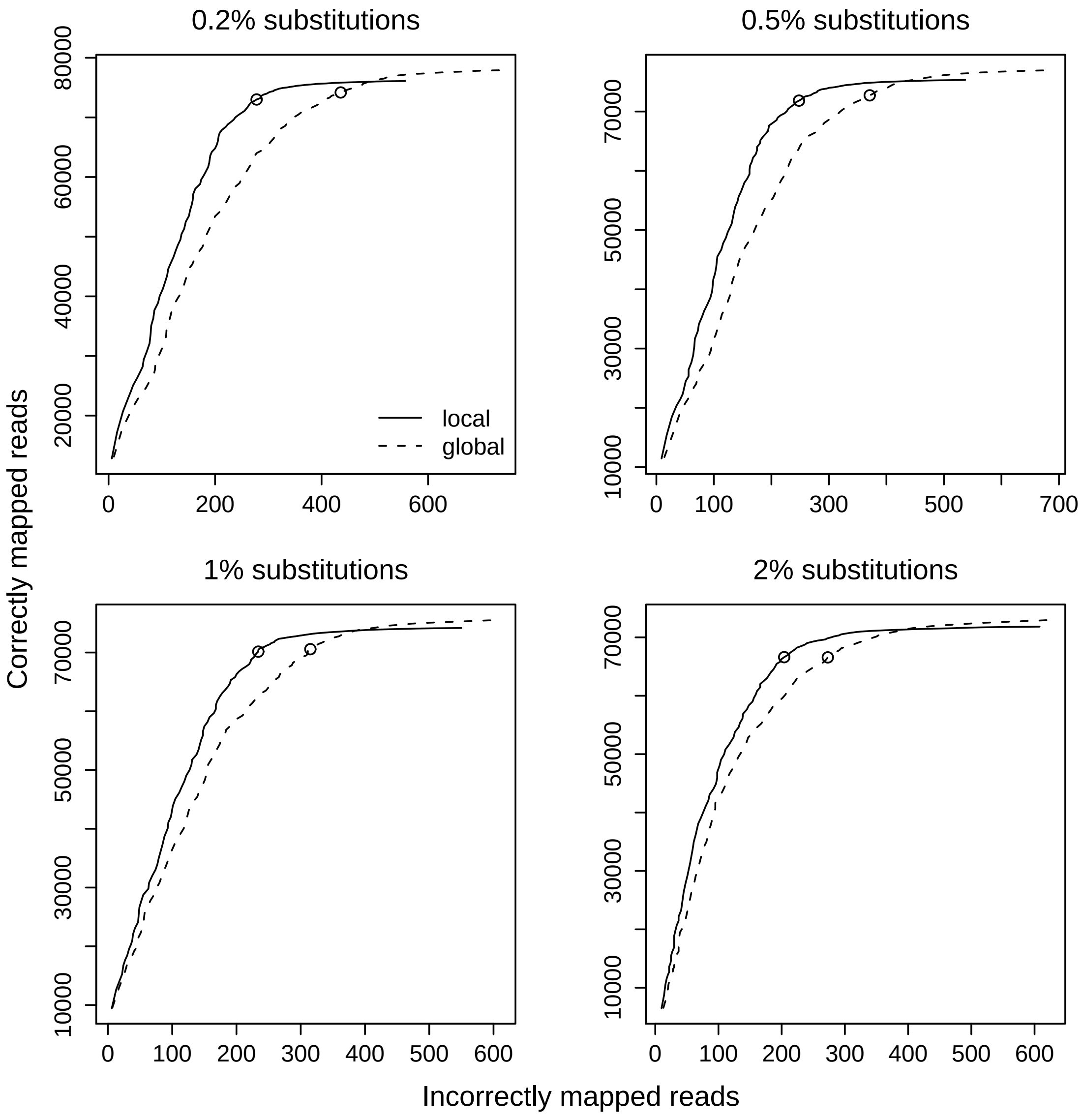

Figure S6: Mapping accuracy for 100,000 simulated 36-bp reads, using either local or semi-global alignment. Here, “semi-global alignment” means that the whole read sequence was forced to align to the genome; “local alignment” means that alignments involving only part of the read were allowed. In all cases, we modeled both the sequencer errors and the “real” substitutions. The other figures all use local alignment: so the solid lines here are identical to those in Figures 2, S2, S4, and S5.

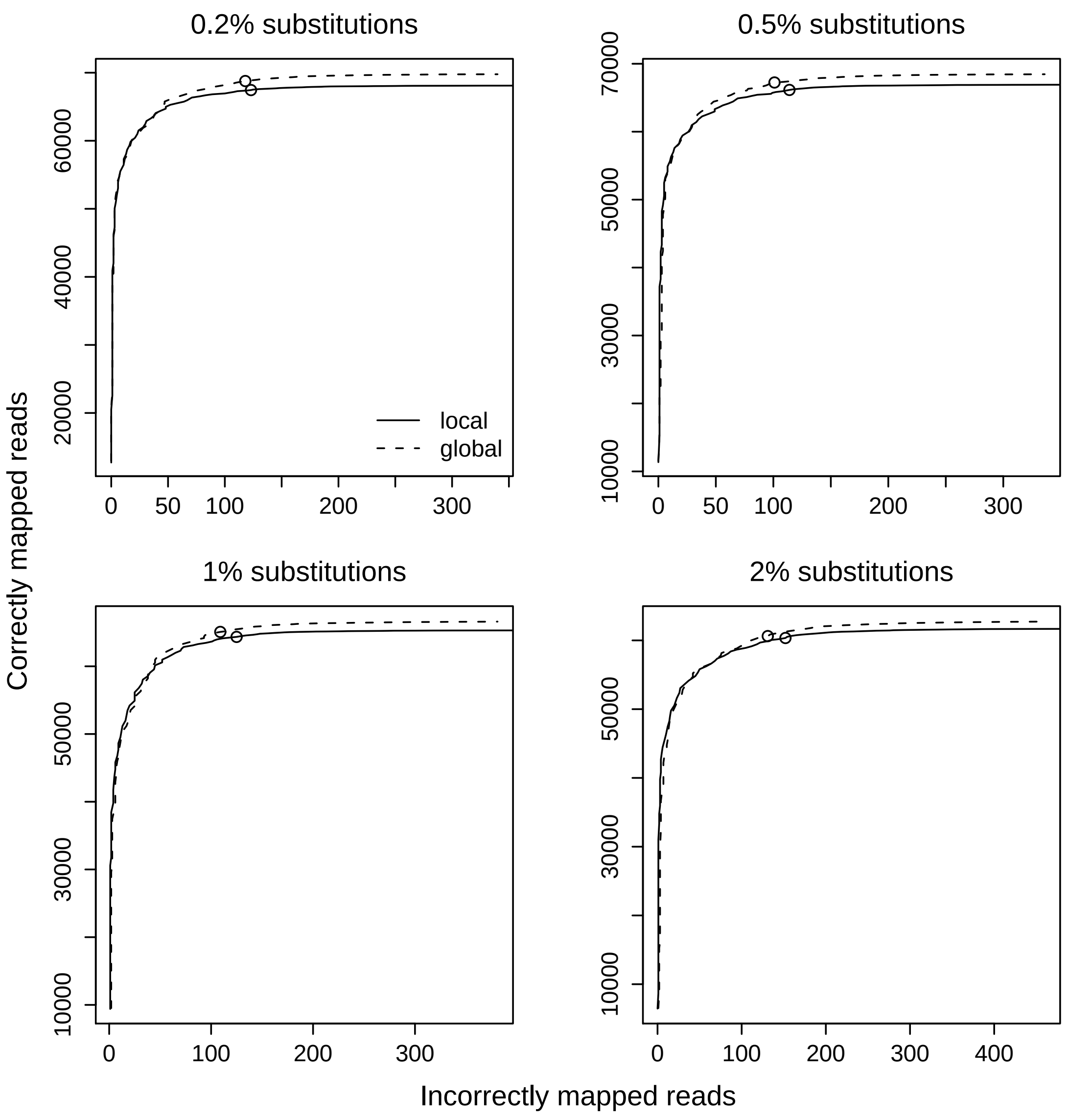

Figure S7: Mapping accuracy for 100,000 simulated 36-bp reads, using either local or semi-global alignment. This is identical to Figure S6, except that here we used the alternative mapping algorithm, which guarantees to find all matches with up to two substitutions. The solid lines here are identical to those in Figure 4 of the main text.

7 Supporting Tables

Table S1: Run times for mapping 100,000 reads with LAST on one core of a 1.86 GHz Xeon E5320 CPU. The execution times exclude the time for indexing the genome, which are: 309 secs for human chromosome 1 in default mode; 307 secs for human chromosome 1 in two-mismatch-guarantee mode; 151 secs for the *D. simulans* genome in default mode; 149 secs for the *D. simulans* genome in two-mismatch-guarantee mode.

	Default mode	Two-mismatch-guarantee mode
Mapping 36-bp reads[1] to human chromosome 1		
Model sequencer errors only	22 sec	130 sec
Model substitutions only	21 sec	185 sec
Model both	30 sec	190 sec
Model both, but ignore the difference between transitions and transversions	23 sec	176 sec
Mapping 51-bp reads[1] to human chromosome 1		
Model sequencer errors only	30 sec	
Model substitutions only	29 sec	
Model both	42 sec	
Model both, but ignore the difference between transitions and transversions	32 sec	
Mapping 36-bp reads to the *D. simulans* genome		
Model sequencer errors only	17 sec	14 sec
Model sequencer errors and real differences, excluding insertions and deletions	24 sec	23 sec
Model sequencer errors and real differences, including insertions and deletions	29 sec	56 sec

[1] Simulated reads with 1% substitutions, plus sequencer errors.

References

[1] S. J. Cokus, S. Feng, X. Zhang, Z. Chen, B. Merriman, C. D. Haudenschild, S. Pradhan, S. F. Nelson, M. Pellegrini, and S. E. Jacobsen. Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning. *Nature*, 452(7184):215–219, March 2008.

[2] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[3] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, November 2008.

[4] B. Ma, J. Tromp, and M. Li. PatternHunter: faster and more sensitive homology search. *Bioinformatics*, 18(3):440–445, March 2002.

[5] Y.-K. Yu, J. C. Wootton, and S. F. Altschul. The compositional adjustment of amino acid substitution matrices. *Proc. National Academy of Sciences USA*, 100(26):15688–15693, December 2003.